(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 124 362 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
16.08.2001 Bulletin 2001/33

(51) Int Cl.7: H04L 29/06

BEST AVAILABLE COPY

(21) Application number: 01300501.2

(22) Date of filing: 19.01.2001

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: 19.01.2000 US 176971 P
27.07.2000 US 626687

(71) Applicants:
- Wiznot Corp.
  Kangnam-ku, Seoul (KR)
- Lee, Jung Tae
  Haewundae-ku, Busan-si (KR)

(72) Inventors:
- Lee, Jung Tae
  Haewundae-ku Busan-si (KR)
- Kim, Ku Hwan
  Donglae-ku, Busan-si (KR)

(74) Representative: Powell, Timothy John
Eric Potter Clarkson,
Park View House,
58 The Ropewalk
Nottingham NG1 5DD (GB)

(54) Apparatus for processing TCP/IP by hardware, and operating method therefor

(57) An apparatus for processing the TCP/IP (Transmission Control Protocol / Internet Protocol) by hardware is disclosed. In this apparatus, an information plane for transmitting, receiving and processing the user data is separated from a control plane for controlling the transmission and reception of the data by using a hardware. Thus the data transmission is carried out regardless of the processing of the control information. Further, a buffer is provided to store the data to be transmitted and the data received, thereby reducing the number of the memory access times. Further, during the transmission of a segment, the calculation of the checksum for the next predictable header and data is carried out simultaneously, thereby reducing the overhead for the checksum calculation. Further, the addresses of the data to be transmitted are inserted into the sequence number field when transmitting the data, while during the reassembling, the values of the sequence number fields are used as the pointers for storing the data into the memory, thereby improving the overhead related to the memory addressing.

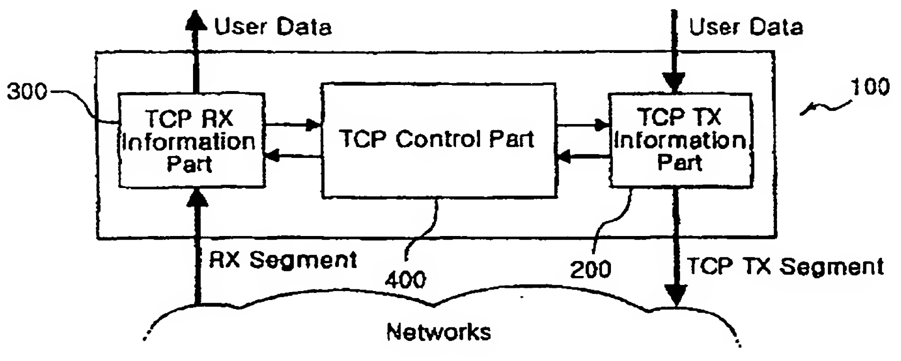


FIG.1

EP 1 124 362 A2

Description

FIELD OF THE INVENTION

[0001] The present invention relates to an apparatus for processing the TCP/IP (Transmission Control Protocol / Internet Protocol) by hardware, and an operating method therefor, in which the user data can be transmitted and received through the internet in the form of TCP segments with an improved efficiency.

DESCRIPTION OF THE PRIOR ART

[0002] If the Internet is to be used, the same communication protocols have to be utilized in the terminals of the both of the concerned parties, and the TCP/IP (Transmission Control Protocol / Internet Protocol) are most generally employed. At present, the TCP/IP is implemented by a software, and are included in the operating system.
[0003] In the TCP, a standardized segment format is used as shown in FIG. 15. Referring to FIG. 15, the source port and the destination port are fields which represent the transmitting side port and the receiving side port for exchanging ɿe data. In the TCP, call connections can be carried out for the respective application programs, and therefore, the ɔlication programs in which the call connections have been carried out are distinguished by utilizing the port number.
ɔ04] Meanwhile, the sequence number field shows the byte position of the transmitted data, which has been sent ɔm the transmitting end. The acknowledgement number field shows the byte position of the data, which is to be received next. The HLEN field shows the length of the header in units of 32 bits. The WINDOW field shows the reserve space of the receiving side buffer. The CODE BIT field consists of 6 bits and is used as shown in table 1. The URGENT POINTER field indicates the ending position of a data in the case where an urgent data is included in the data field. The option field is for adding various options, and the kinds of the options are very diversified as illustrated in FIG. 16.

Table 1

Code bit	In case set to 1
URG	URGENT POINTER field is effective.
ACK	Acknowledgement field is effective.
PSH	The data which has been sent is directly transferred to the user without buffering.
SYN	Requests or allows call connections.
FIN	Requests or allows connection release.
RESET	Initializes the connection setting.

[0005] As shown in FIG. 16, the option field consists of Kind, Length and Option data. In FIG. 16, "End of Option List" shows the end of option, and "No Operation" is used for padding the respective options in 4-byte units. Further, "Maximum Segment Size option" is used for determining the maximum segment size (MSS) carrying out the transmissions and receptions while connecting call. Further, "Window Scale Factor" is used for an option for expanding the window field of the presently used TCP, and "Timestamp Option" is used for calculating the Round Trip Time(RTT) by exchanging the time information between the transmitting and receiving sides.
[0006] Referring to FIG. 15, "The Checksum Field" is for checking the transmission errors for the header and the data. Under this condition, the checksum calculation is carried out by making a pseudo header of FIG. 17 involved. That is, the TCP checksum calculates one final checksum by unifying the data checksum, the TCP header checksum and the pseudo header checksum. In the checksum calculation by the TCP, all the data are summed up in 16-bit units, and the complementary number of 1 is taken. Such calculations are carried out for the header and the data in the TCP, while in the IP, the calculation is carried out only for the header.
[0007] Meanwhile, as shown in FIG. 17, the pseudo header includes: the IP address of the transmitting end, the IP address of the receiving end, the protocol field for showing the values of the upper level protocols, and the TCP length field. For example, in the case where a TCP segment is transmitted at the IP layer, the protocol field is set to 6, while the TCP length is the sum addition of the TCP header and data.
[0008] The TCP which is presently used in the internet consists of a software. However, this conventional software TCP cannot transfer the efficiency from the lower level to the user application, first due to the overhead of the structural problems, second due to the overhead of the memory access problems, and third due to the overhead of the checksum calculating problems.
[0009] To describe the above problems in more detail, first in the conventional TCP software, the control information and the data information are handled without being distinguished from each other, and therefore, the TCP processing time is not constant due to the processing of the control information even during the data transmission, as well as the

performance being lowered. Particularly, in the multi-media application in which a predetermined processing time has to be ensured, the above cited demerits can be fatal.

[0010] Second, as shown in Table 2 below, when the TCP is embodied, the time of transferring the memory data to the network card or the time of transferring the received data (received into the network card) to the user region of the memory is very long. Accordingly, as the number of times of the memory access becomes large, so much the overhead is increased.

Table 2.

Time of processing the TCP protocol	
Items processed	Processing time [μs]
User - system copy	200
TCP checksum	185
Network - memory copy	386
Ethernet drivers	100
TCP+IP+ARP protocols	100
Operation system overhead	240

[0011] In the conventional embodying means as shown in FIG. 18, when the data is transmitted, the data which has been stored in the user region of the memory 820 is moved to a kernel buffer (①,②), then a CPU 810 reads the data of the kernel region of the memory 820 to calculate the checksum (③), and then, the data of the kernel buffer is transferred to a network card 830 based on a DMA method (④). In the same manner, when the data is received, first the received data is stored in the network card 830, then the data is transferred to the kernel buffer (④), then the CPU 810 reads the data to check any errors (③), and then, if there is no error, the data is transferred from the kernel buffer to the user buffer (②,①). That is, for any one of data, four accesses are carried out, and therefore, the performance of the overhead becomes problematic.

[0012] Third, in the TCP as shown in FIG. 15, the checksum field is positioned at the header, and therefore, in the conventional checksum design method, the relevant segment can be transmitted only after the completion of the calculation of the checksum. For example, in the case where a data of 1460 bytes is transmitted, first the 20 bytes of the TCP header data and the 12 bytes of the pseudo header data are read to calculate the checksum, and then, the 1460 bytes of the data are read to calculate the checksum. Only then, the TCP segment can be transmitted. Accordingly, during the time when the checksums are calculated, the data cannot be transmitted, and therefore, so much a time delay occurs in carrying out the data transmission. In the case of the IP protocol, the checksum calculation gives no much problem, because, in this case, the header size is generally 20 bytes, and 10 command codes will be sufficient if the calculation is carried out at 16 bits. However, in the TCP protocol, the checksums not only for the TCP header but also for the total data have to be calculated, and therefore, the overhead is excessively increased.

[0013] Fourth, when the TCP transmits the data of application layers, the data is cut into segments of a certain size at the transmitting end, and the data is reassembled at the receiving end. For this, at the receiving end, a link list type data structure is formed to store the data in the memory, so that the data positions would not be wrongly located. In this conventional method, the reassembling of the data is carried out after the completion of the reception of the data, and therefore, the overhead becomes problematic.

SUMMARY OF THE INVENTION

[0014] The present invention is intended to overcome the above described disadvantages of the conventional technique.

[0015] Therefore, it is an object of the present invention to provide an apparatus for processing the TCP/IP by hardware, and an operating method therefor, in which the user data can be transmitted and received through the internet in the form of TCP segments with an improved efficiency.

[0016] In achieving the above object, the apparatus and method according to the present invention are characterized as follows. That is, in the TCP/IP, an information plane for transmitting, receiving and processing the user data is separated from a control plane for controlling the transmission and reception of the data by using a hardware. Thus the data transmission is carried out regardless of the processing of the control information. Further, a buffer is provided to store the data to be transmitted and the data received, thereby reducing the number of the memory access times. Further, during the transmission of a segment, the calculation of the checksum for the next predictable header and data is carried out simultaneously, thereby reducing the overhead for the checksum calculation. Further, the addresses of the data to be transmitted are inserted into the sequence number field when transmitting the data, while during the reassembling, the values of the sequence number fields are used as the pointers for storing the data into the memory, thereby improving the overhead related to the memory addressing.

BRIEF DESCRIPTION OF THE DRAWINGS

[0017] The above object and other advantages of the present invention will become more apparent by describing in detail the preferred embodiment of the present invention with reference to the attached drawings in which:

FIG. 1 is a block diagram showing the overall constitution of the apparatus for processing the TCP/IP by hardware according to the present invention;
FIG. 2 is a flow chart showing the operation of the apparatus for processing the TCP/IP by hardware according to the present invention;
FIG. 3 is a block diagram showing the TX information part of the apparatus for processing the TCP/IP by hardware according to the present invention;
FIG. 4 is a block diagram showing the TCP RX information part of the apparatus for processing the TCP/IP by hardware according to the present invention;
FIG. 5 is a block diagram showing the control part of the apparatus for processing the TCP/IP by hardware according to the present invention;
FIG. 6 is a block diagram showing the buffer management in the apparatus for processing the TCP/IP by hardware according to the present invention;
FIG. 7 is a block diagram showing the memory access method in the apparatus for processing the TCP/IP by hardware according to the present invention;
FIG. 8 illustrates a part of the constitution of the apparatus for processing the TCP/IP by hardware according to the present invention, with a pipeline checksum calculation method being applied;
FIG. 9 is a flow chart showing the sequence of the pipeline checksum calculation method according to an embodiment of the present invention;
FIG. 10 is a timing diagram showing a size of transmission data varied as a time in a multi-channel environment.
FIG. 11 is timing diagram describing the checksum calculation method according to other embodiment of the present invention.
FIG. 12 is flow chart showing the sequence of the pipeline checksum calculation method according to other embodiment of the present invention.
FIG. 13 illustrates a transmission method using a sequence number in the apparatus for processing the TCP/IP by hardware according to the present invention;
FIG. 14 illustrates the reassembling of the data by using the sequence numbers in the apparatus according to the present invention;
FIG. 15 illustrates the structure of the segment based on the TCP (transmission Control Protocol);
FIG. 16 illustrates the kinds of the option field in the TCP segment;
FIG. 17 illustrates the structure of the pseudo header;
FIG. 18 is a block diagram showing the conventional memory access method; and
FIG. 19 comparatively illustrates the conventional checksum calculation method and the checksum calculation method according to the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0018] The apparatus for processing the TCP/IP (Transmission Control Protocol / Internet Protocol) by hardware, and an operating method therefor, will be described below. In this context, the part of processing the TCP is same as that of processing the IP, and therefore, only the part of processing the TCP will be described.
First, the present invention solves the overhead structural problems caused by the simultaneous processing of the control information and the data information. For this purpose, a control part for carrying out the control processing is separated from an information part for carrying out the data transmission and reception. One example of this feature is illustrated in FIG. 1.
[0019] FIG. 1 is a block diagram showing the overall constitution of the apparatus for processing the TCP/IP by hardware according to the present invention. Particularly, FIG. 1 illustrates a part of processing the TCP. As shown in this drawing, the apparatus 100 according to the present invention includes: a TX information part 200 for processing a user data to transmit it to a destination; an RX information part 300 for receiving a TCP segment from a transmitting side to process it, and to transmit a user data to an application layer; and a control part 400 for processing the control information related to the transmission and reception data.
[0020] As described above, the data processing part is separated from the control information processing part in the

present invention. The TX information part 200 and the RX information part 300 handle the transfer of the information, while the control part 400 manages the protocol status so as to control the operations of the information parts 200 and 300. Therefore, as long as a sudden event such as an error or an overflow does not occur, the information parts 200 and 300 can transmit the user information without being concerned with the control information.

[0021] FIG. 2 is a flowchart showing the operation of the apparatus for processing the TCP/IP by hardware according to the present invention. If a command (API command) is received from an application layer (s201), then the control part 400 receives it to take a proper action (s202). If a user data or a TCP segment is received (s203), then a checking is made as to whether there is present a control information such as an error, a timeout, a SYN or a FIN (s204 - s206). If there is a control information, the control part 400 processes it, while the data is processed by the information parts 200 and 300 (s207).

[0022] Now the respective parts of the apparatus of FIG. 1 will be described as to their constitutions and actions.

[0023] FIG. 3 illustrates the TX information part 200 which forms the header of the TCP segment, and transfers it to the network layer together with the user data.

[0024] The TX information part 200 includes: a header register 210 for storing a received TCP header information; an HLGM (header length generating module) 220 for receiving a header information from the header register 210 to calculate a length of the header; a TLGM (total length generating module) 230 for receiving a data length information from the header register 210, and for receiving a header length value from the HLGM 220, to calculate a total segment length as the sum of the header length and the data length; a PCGM (partial checksum generating module) 240 for receiving a part of the TCP header and a data, to calculate a partial checksum for a predictable values among the TCP headers and the data; an FCGM (final checksum generating module) 250 for adding a window value and an ACK number (decided immediately before a transmission of the TCP segment) to the partial checksum of the PCGM (240), to generate a final checksum; a TX buffer 260 for receiving a user data (to be transmitted) through a bus of a host system, to store it; a temp buffer 270 for temporarily storing the data (which has been read to calculate the checksum); an output header buffer 280 for temporarily storing the header information together the final checksum of the FCGM 250; and a TX Mux (TX multiplexer) 290 for summing up the user data of the temp buffer 270 and the checksum of the output header buffer 280, to generate a TCP segment.

[0025] The header of the TCP segment basically has 20 bytes, but its length can be varied depending on the kind of the option field. Therefore, each time when one segment is transmitted, the length of the header has to be known. The input of the HLGM 220 is the signals which indicate the presence or absence of the option which is provided by the control part 400. The output of the HLGM 220 is the signals which indicate the header length.

[0026] Meanwhile, if the checksum calculation which is one of the functions of the TX information part 200 is to be carried out, not only the header and data as described above but also the pseudo header of FIG. 17 are required. The pseudo header includes the IP addresses of the transmitting and receiving sides, the kinds of the protocols, and the lengths of the TCP. Of them, the length of the TCP is calculated by the TLGM 230. The input of the TLGM 230 includes the header length value as the output of the HLGM 220 and the length value of the data to be transmitted. The output of the TLGM 230 includes the TCP length of the TCP segment, and this output is inputted into the FCGM 250.

[0027] Referring to FIG. 3, the temp buffer 270 which temporarily stores the calculated checksum data is an FIFO (first-in first-out) type buffer.

[0028] FIG. 4 is a block diagram showing the RX information part 300 of the apparatus for processing the TCP/IP by hardware according to the present invention. As shown in this drawing, the RX information part 300 includes: an RX Demux (RX demultiplexer) 310 for separating the header and the data from each other in the received TCP to output them; an HCCM (header checksum control module) (320) for checking the checksum of the header (separately outputted from the RX Demux 310) to judge on transmission errors; a DCCM (data checksum control module) 330 for checking the checksum of the data (separately outputted from the RX Demux 310) to judge on transmission errors; an RX header processing module 340 for carrying out a relevant processing in accordance with the output header information of the RX Demux 320, and for discarding the received data if an error is found in the header or the data by the HCCM 320 or the DCCM 330, to carry out an operation based on the received header; and a header register 350 for storing the received header.

[0029] In the above, a part of the header which is stored in the header register 350 is transferred to the control part 400.

[0030] FIG. 5 illustrates the control part 400 which generates proper control signals in accordance with the status of the protocol so as to control the information parts 200 and 300. As shown in this drawing, the control part 400 includes: a TX control module 410 for controlling the TX control part 200 in accordance with the status of the protocol; an RX control module 420 for generating proper control signals in accordance with the status of the protocol to receive the TCP segment (directed toward itself) so as to control the RX information part 300; and a TCP control module 430 for controlling the entire apparatus 100 to process the TCP by hardware.

[0031] For example, if a congestion occurs in the network, the TX control module 410 generates a congestion window value to adjust the amount of the transmitting data. Further, the module 410 carries out the functions of a time-out, an RTT generation, and a congestion manager. If an error occurs in the transmitted data, the module 410 orders a transmission of the relevant data to the TX information part 200.

[0032] The RX control module 420 checks as to whether the destination port of the received segment is correct, and controls the pointer of the buffer to manage an RX buffer 360 of the RX information part 300. That is, FIG. 6 illustrates the RX buffer 360 and a counter which is used for managing the mentioned buffer. Here, a WC (write counter) is a pointer which indicates the position for inserting the received user data. An RC (read counter) is a pointer which indicates the starting position for the data to be read and to be transmitted. An FC (free counter) is a pointer which indicates the position for receiving an acknowledgement after the transmission of the data. The WC, the RC and the FC all perform a circular queue type operation in which they return to the original positions upon encountering the boundary. In FIG. 6, the region (A) indicates the stored user data which has not been transmitted. The region (B) indicates the data which has been transmitted but for which an acknowledgement has not been received. The region (C) indicates a free region in which the acknowledgement has been received, and therefore, the region is vacant.

[0033] The TCP control module 430 controls various command registers within the TCP generating apparatus 100 to process the commands of the user API. Further, the TCP control module 430 controls control registers and flag registers to order the operations of the TX/RX control modules 410 and 420. Further, the TCP control module 430 controls status registers to notify the status of the TCP generating apparatus 100 to the user API. Further, it manages an FSM (finite state machine), and sets or releases connections.

[0034] Second, in the present invention, the overhead due to the memory access is improved. For this purpose, exclusive buffers for storing the data are provided within the apparatus 100 so as to reduce the number of times of memory access. That is, there are provided a TX buffer 260 within the TX information part 200, and an RX buffer 360 within the RX information part 300. Further, in the apparatus 100, the checksum calculation is not carried out in the CPU of the host system, but is carried out in the apparatus 100, and therefore, the memory access times are decreased, thereby improving the overhead to a significant degree.

[0035] As shown in FIG. 7, it will be assumed that: reference code 740 indicates the memory of the host system, 730 indicates the CPU of the host system, 720 indicates the bus, and 710 indicates the apparatus for processing the TCP by hardware. Then the data which is stored in the user region of the memory 740 is directly transmitted to the buffer 711 of the TCP generating apparatus 710. Further, the data which is stored in the buffer 711 is directly transmitted to the user region of the memory 740.

[0036] Third, in order to improve the overhead during the checksum calculation in the present invention, when transmitting a segment based on a pipe line technique, the checksum of a predictable field of the next transmitting segment is calculated simultaneously. The checksum calculation method in the apparatus 100 of the present invention can be categorized into checksum calculation method for static data length and checksum calculation method for variable data length.

[0037] FIG. 8 is a block diagram specifically showing the checksum calculation for static length data in the apparatus 100 of the present invention. That is, the data to be transmitted is stored in the TX buffer 260, and the PCGM 240 receives the data of the TX buffer 260 and a predictable TCP header. Then the checksum of the inputted data and the TCP header is calculated, thereby generating a partial checksum. Here, the predictable TCP header corresponds to all the fields of FIG. 15 excluding the acknowledgement number, the WINDOW, and the code bits. That is, the mentioned "all the fields" are those which are not changed during the time when the TCP connection is maintained.

[0038] The partial checksum which has been generated in the above described manner is inputted into the FCGM 250. and the FCGM 250 receives the ACK number, the WINDOW value and the code bits which are decided immediately before the transmission, thereby generating a final checksum. During the time when the checksum is calculated, the relevant user data is stored in the FIFO type temp buffer 270. The procedure of reading the data of the temp buffer 270 and the procedure of calculating the checksum for the data to be transmitted next, are simultaneously carried out.

[0039] As described above, the fact that the checksum for the predictable header and the data can be calculated owes to the fact that in order to calculate the checksum by the Transmission Control Protocol (TCP), the input values are summed in 16-bit units, and then, a complementary number of 1 is taken so as to add them.

[0040] The operation algorithm by which the checksum is calculated by the TX information part 200 based on the pipe line method corresponds to the flow chart of FIG. 9.

[0041] As shown in this drawing, first a checking is made as to whether a data to be transmitted is present in the memory (s901). If there is one, then a checking is made as to whether it is a first transmitting data (s902). Under this condition, if it is a first transmitting data, there is no calculated checksum value. Therefore, a checksum is calculated (s903), and then, a TCP segment is generated to transmit the data (s904). Under this condition, when the data is transmitted, a checksum for the next transmitting data and header is simultaneously calculated (s905). Under this condition, the data which is read for calculating the checksum is stored in the temp buffer 270. On the other hand, if the data is not a first transmitting data, there should be a checksum which has been already calculated at the preceding step. Therefore, a segment is formed by using the already calculated checksum and the header and the data (which are stored in the temp buffer 270) to transmit the data (s906). At the same time, a checksum for the next transmitting data and header is calculated (s907).

[0042] As described above, in the present invention, in order to reduce the overhead caused by the calculation of the checksum of the TCP, the checksum for the predictable data region among the fields of the TCP header and the pseudo header is calculated in advance. Thus the pipe line technique is adopted in which the transmission procedure and the checksum calculation procedure are overlapped in time. In this context, the fields of the TCP header and the pseudo header include: source port, destination port, header length, code bits, urgent pointer, MSS, protocol, source IP address, destination IP address, and TCP length. As to the data also, the checksum can be calculated in advance by reading the data from the buffer in accordance with the max segment size value which is decided during the call connection.

[0043] Accordingly, the method can be compared with the conventional method as follows. That is, in the conventional method as shown in FIG. 19a, after the checksum for the first data and header is formed, the segment consisting of the first data and header is transmitted. In contrast to this, in the method of present invention as shown in FIG. 19b, the transmission of a segment is carried out simultaneously with the calculation of the checksum for the next transmitting segment. Therefore, if it is assumed that three segments are to be transmitted, then 6 units of time are required in the conventional method, whereas in the present invention, only 4 units of time are consumed. This time difference is so much increased, as the amount of the data to be transmitted is increased. In the specific terms, usually twice or more of time is required in the conventional method compared with the present invention. In the Fig 19a, b, The numbers illustrated represent a unit of time, and the data size is all the same.

[0044] Said pipeline-type checksum calculation method is suitable for a mono channel environment, and it is possible to apply when the size of data to generate is all the same or when a buffer is filled with data to be transmitted. But in case of multi-channel environment, TCP hardware processes data in a channel and then processes data in another channel. Therefore, in view of processing data in a channel, data process and transmission will happen in a periodical manner. But, in view of apparatus, as shown in FIG 10, data to be transmitted will be varied as time.

[0045] In FIG 10, t0 illustrates time that an application starts to generate data, according to the apparatus of the present invention, data (D1) will be processed for the first time at t1 when is 8t after other data is being processed. As shown in FIG 10, data to be transmitted during t1 is 1460 bytes, t2 is 500 bytes, t3 is 200 bytes, t4 is 1460 bytes and t5 is 500 bytes, which represent different size of data to be processed.

[0046] Therefore, checksum calculation method for variable length data is preferred in the present invention.

[0047] FIG 11 illustrates a timing diagram showing pipeline-type checksum calculation method, and FIG 12 represents a flow chart showing the process of pipeline-type checksum calculation method for processing said variable length data

[0048] As shown in FIG 11 and FIG 12, when the length of checksum data changes, all the channels will be on stand-by status after starting operation (s1201). And then all the channels will check if it is time to transmit data (t*n*). And if the time is t*n* to transmit data at a channel *n*, send data at the channel (s1205), wherein the transmitted data is stored data after checksum calculation completes.

[0049] When the time is not t***n*** to transmit data at a channel ***n***, check if data to be processed exists (s1203). If data to be processed exists, as stated above, starts checksum calculation to send data at time t***n***. (s1204)

[0050] Said process is preceded for each channel, and pipeline-type checksum calculation can be done in advance for any time period during 8t, therefore data transmission capacity can be maximized up to 2.

[0051] As described above, the calculation of the checksum is carried out by hardware, and therefore, the calculation speed is improved compared with the conventional method in which the calculation is carried out by software. Further, the data which has been read from the memory for the calculation of the checksum is stored in the FIFO type buffer, and when the transmission is carried out, the data of the buffer can be transmitted directly, thereby reducing the number of the memory access times.

[0052] The conventional TCP software is operated in the kernel, and the OS was responsible for the task of storing and managing the data. In contrast to this, in the present invention, a buffer managing method using a sequence number is adopted, so that the overhead related to the buffer allocations and to the restoration and management of the buffers can be reduced. The sequence number is one of the fields of FIG. 15. That is, for the data which is being transmitted, a number is assigned to each byte unit, and the first byte number of the TCP segment is stored into the sequence number field. In the present invention, the receiving end carries out the reassembling of the data by using the sequence number, or detects any omitted data by using the sequence number, thereby requesting for a retransmission.

[0053] That is, by using the sequence number as the address of the buffer, the procedure of transmitting the data, the procedure of retransmitting the data and the procedure of reassembling the data are carried out in a simple manner. Further, three pointers such as WC, RC and FC are employed as shown in FIG. 6. The user data which has been received from the application is stored in the address which is indicated by the WC pointer. When transmitting the data, the data which is stored in the address indicated by the RC pointer is read and transmitted. If a retransmission is requested, the data of the address which is indicated by the FC pointer is copied to store it into the address indicated by the RC pointer. Then the data of the address which is indicated by the RC pointer is read and transmitted.

[0054] FIG. 13 illustrates a buffer managing method in which the data is transmitted by using the sequence number. FIG. 14 illustrates the reassembling of the data after receipt of the data.

[0055] When transmitting a data as shown in FIG. 13, a WC pointer value which indicates a physical address in the memory is inserted into the sequence number field of the segment to be transmitted. The data of the address which is indicated by the WC pointer is inserted as the data to be transmitted.

[0056] In FIG. 14, when reassembling the received data, the value of the sequence number field for the received segment is stored into the buffer based on the WC pointer value, and the data of the received segment is stored into the address which is indicated by the WC pointer.

Claims

1. An apparatus for processing TCP/IP (Transmission Control Protocol / Internet Protocol) by hardware, comprising:

    an information plane for processing a user data to be transmitted; and
    a control plane for carrying out controls in connection with data transmission and reception, said two planes being separately provided.

2. The apparatus as claimed in claim 1, further comprising exclusive buffers for storing a data to be transmitted and a data to be received, so as to make the data (to be transmitted and received) directly transferred between a memory of a host system and said apparatus for processing the TCP/IP.

3. An apparatus for processing TCP/IP by hardware, comprising:

    a TX information part for processing a user data to transmit it to a destination;
    an RX information part for receiving a TCP segment from a transmitting side to process it, and to transmit a user data to an application layer; and
    a control part for processing the control information related to the transmission and reception data, whereby a data transmission or reception is carried out regardless of a processing of a control information.

4. The apparatus as claimed in claim 3, further comprising exclusive buffers for storing a data to be transmitted and a data to be received, so as to directly transfer a transmitting data from a memory of a host system to said exclusive buffer, and so as to directly transfer a receiving data to said memory of said host system.

5. The apparatus as claimed in claim 3, wherein said TX information part comprises:

    a header register for storing a received TCP header information;
    an HLGM (header length generating module) for receiving a header information from said header register to calculate a length of a header;
    a TLGM (total length generating module) for receiving a data length information from said header register, and for receiving a header length value from said HLGM, to calculate a total segment length as a sum of the header length and the data length;.
    a TX buffer for receiving a user data (to be transmitted) through a bus of a host system, to store it;
    a PCGM (partial checksum generating module) for calculating a checksum for predictable header and data before a transmission to generate a partial checksum;
    an FCGM (final checksum generating module) for receiving a header value (decided immediately before a transmission of a TCP segment) and the partial checksum of said PCGM, to generate a final checksum;
    an output header buffer for temporarily storing the header information together the final checksum of said FCGM;
    a temp buffer for receiving the data (which has been read to calculate the partial checksum by said PCGM) to temporarily store the data; and
    a TX Mux (TX multiplexer) for receiving the header information and the data from said output header buffer and said temp buffer to generate a TCP segment.

6. The apparatus as claimed in claim 3, wherein said RX information part comprises:

    an RX Demux (RX demultiplexer) for separating the header and the data from each other in a received TCP segment;

an HCCM (header checksum control module) for checking the checksum of the header (which has been separately outputted from said RX Demux) to judge on transmission errors;
a DCCM (data checksum control module) for checking the checksum of the data (which has been separately outputted from said RX Demux) to judge on transmission errors;
an RX buffer for storing a data after its separate outputting from said RX Demux;
an RX header processing module for carrying out a relevant processing in accordance with the output header information of said RX Demux, and for discarding the received data by controlling said RX buffer if an error is found by said HCCM or said DCCM in the header or the data; and
a header register for storing the received header.

7. The apparatus as claimed in claim 3, wherein said control part comprises:

a TX control module for generating proper control signals to control said TX control part in accordance with the status of the protocol, and for ordering a retransmission to said TX information part upon encountering an error in a transmitted segment;
an RX control module for generating proper control signals in accordance with the status of the protocol to control said RX information part, and for checking as to whether a destination port of a received TCP segment is correct; and
a TCP control module for processing commands incoming from a user API, for controlling operations of said TX control module and said RX control module, and for notifying operation statuses of said modules to a user application.

8. The apparatus as claimed in any one of claims 5 to 7, wherein said apparatus comprises:

a WC (write counter) as a pointer which indicates a position for inserting a received user data;
an RC (read counter) as a pointer which indicates a starting position for a data to be read and to be transmitted; and
an FC (free counter) is a pointer which indicates a position for receiving an acknowledgement after a transmission of a data.

9. An apparatus for processing TCP/IP by hardware, comprising a checksum calculating structure,
said checksum calculating structure comprising:

a first module for calculating a partial checksum for predictable header value and data before carrying out a transmission;
a second module for receiving the partial checksum of said first module and a header value (which is decided immediately before the transmission) to generate a final checksum; and
a temp buffer for temporarily storing a user data until the final checksum is generated through said first module and by said second module.

10. An operating method of an apparatus for processing TCP/IP by hardware, comprising the steps of:

checking as to whether there is a data to be transmitted;
checking as to whether the data is a first transmitting data if there is found a data to be transmitted;
calculating a checksum if the data is a first transmitting data;
generating a TCP segment to transmit it, and simultaneously calculating a checksum for a next transmitting data and a header; and
inserting the checksum and data of the preceding step to generate and transmit a TCP segment if the mentioned data is not a first transmitting data, and simultaneously calculating a next transmitting data and a header.

11. An operating method of an apparatus for processing TCP/IP by hardware, comprising the steps of:

checking if it is time to transmit data on stand-by status;
transmitting data if it is time to transmit data;
checking if data exists when it is not the time to transmit data;
and starting ckecksum calculation for transmitted data, when it is not the time of sending data;

and said steps are operation method of transmission control protocol creation device characterized by being processed for each channel.

12. An operating method of an apparatus for processing TCP/IP by hardware, comprising the steps of:

inserting a buffer address of a transmitting data into a sequence number field, and inserting a data of a relevant address into a data field, to generate respective segments, when transmitting the data; and
storing the data of the received segment into an address of a buffer (which corresponds to a value of the sequence number field) when reassembling the data of a received segment.

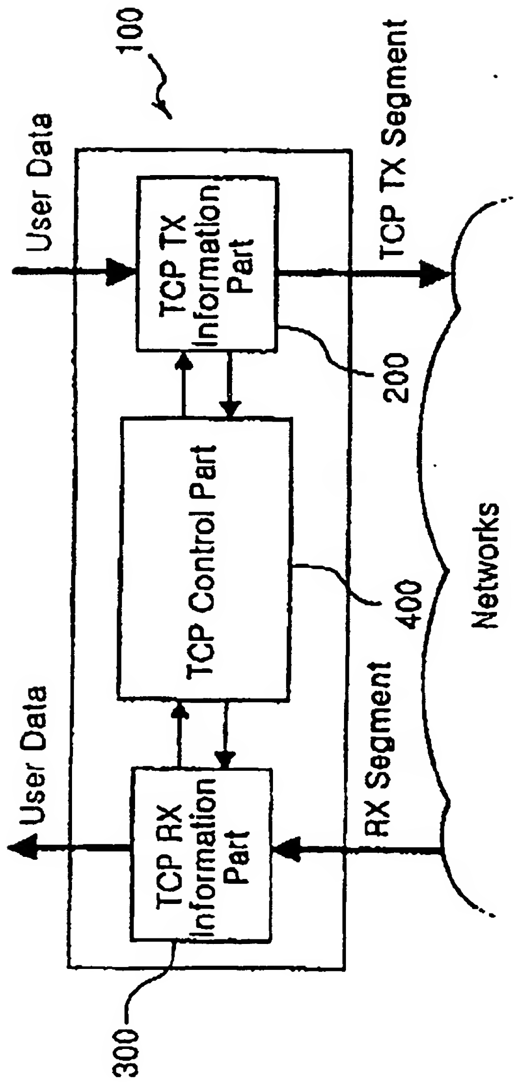

100
User Data
TCP TX Information Part
TCP TX Segment
200
TCP Control Part
400
Networks
User Data
TCP RX Information Part
RX Segment
300

FIG.1

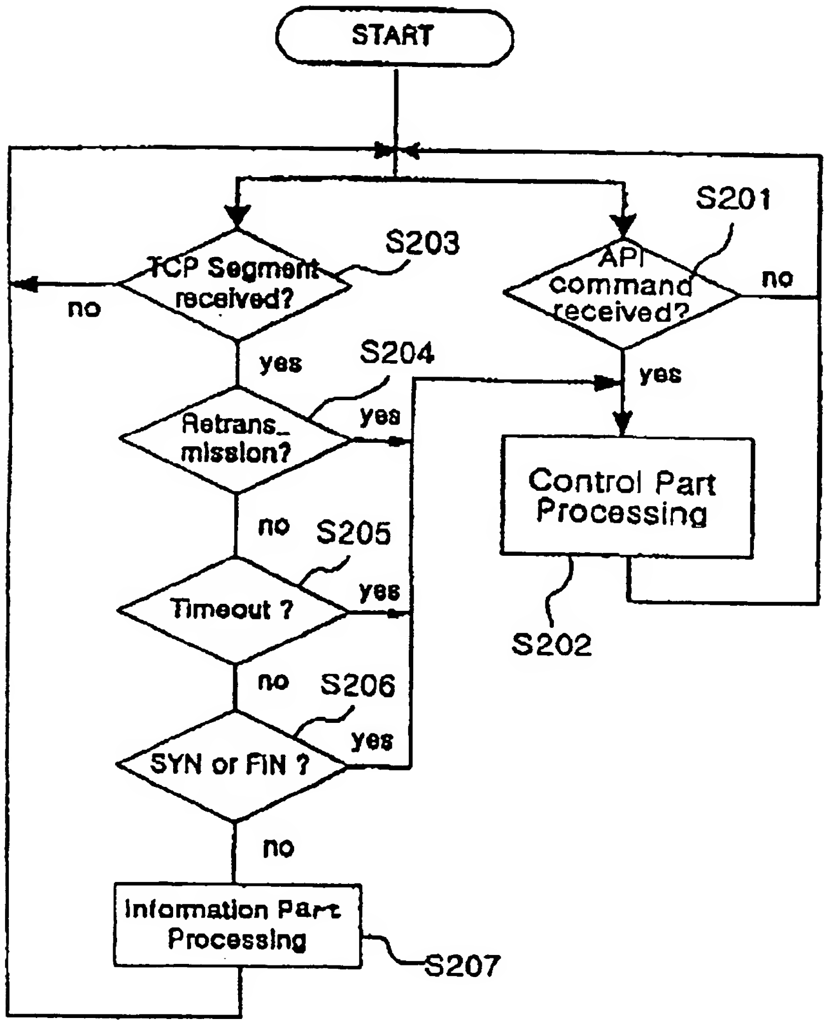

START
TCP Segment received?
S203
no
yes
S204
Retrans_ mission?
yes
no
S205
Timeout ?
yes
no
S206
SYN or FIN ?
yes
no
Information Part Processing
S207
API command received?
S201
no
yes
Control Part Processing
S202

FIG.2

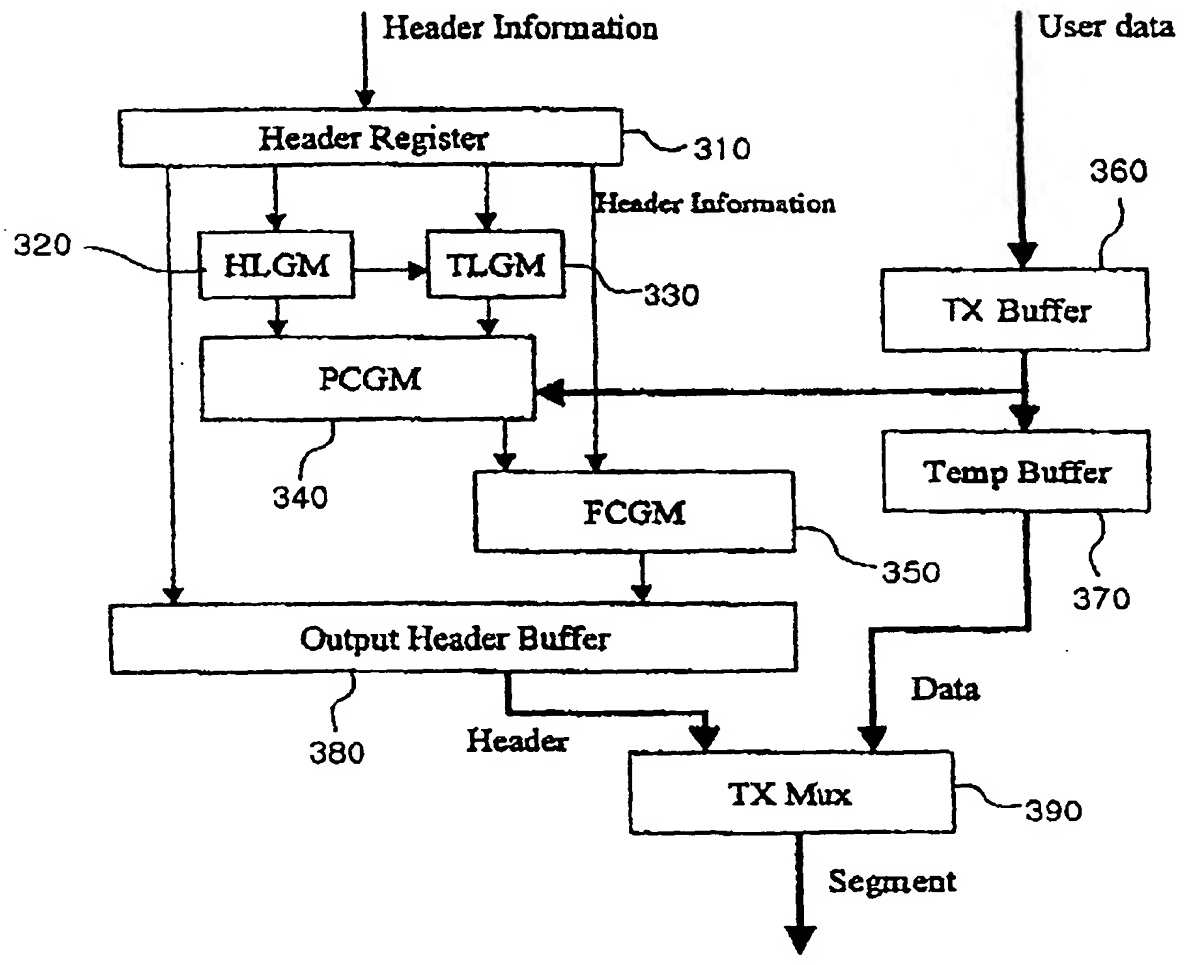

Header Information
User data
Header Register
310
Header Information
360
320
HLGM
TLGM
330
TX Buffer
PCGM
Temp Buffer
340
FCGM
350
370
Output Header Buffer
Data
380
Header
TX Mux
390
Segment

FIG.3

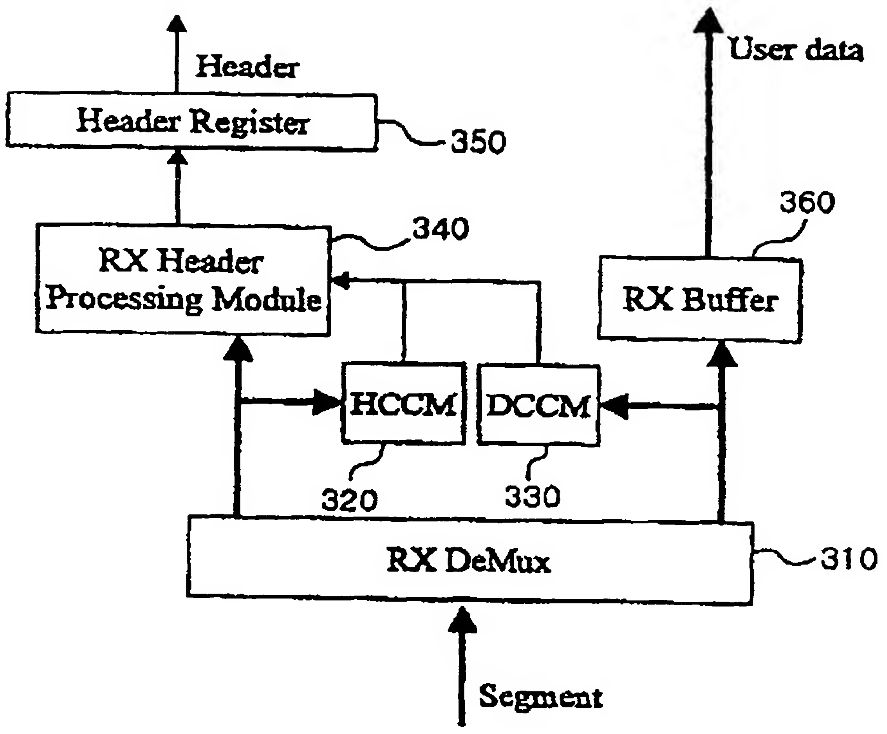

Header
Header Register
350
User data
360
340
RX Header Processing Module
RX Buffer
HCCM
DCCM
320
330
RX DeMux
310
Segment

FIG.4

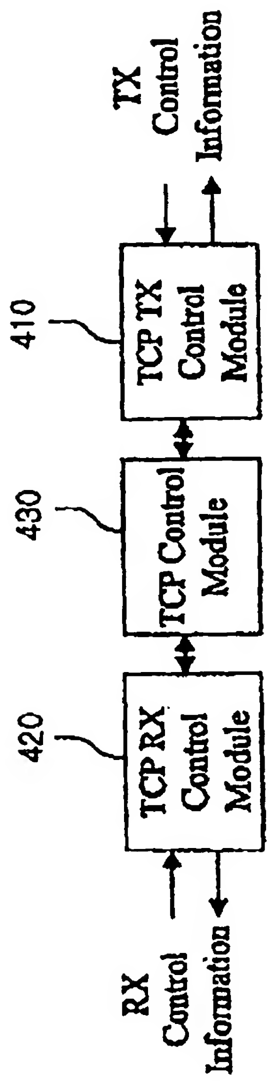

RX Control Information
420
TCP RX Control Module
430
TCP Control Module
410
TCP TX Control Module
TX Control Information

FIG.5

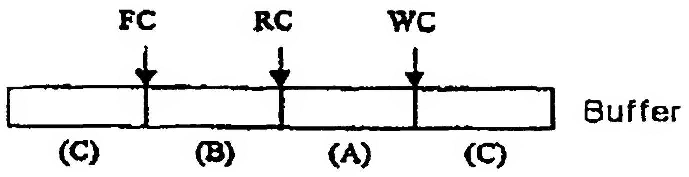


FIG.6

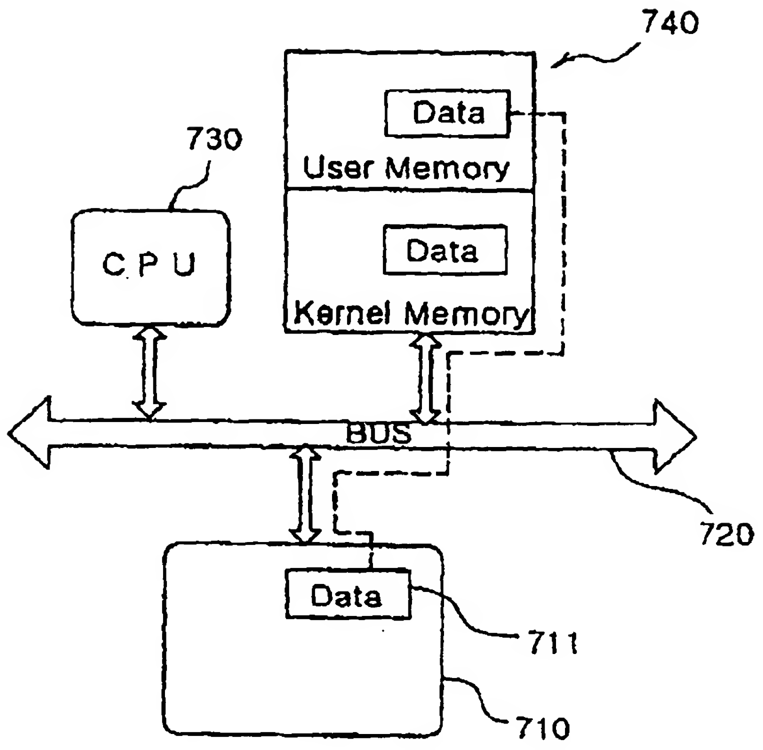

740
Data
User Memory
730
CPU
Data
Kernel Memory
BUS
720
Data
711
710

FIG.7

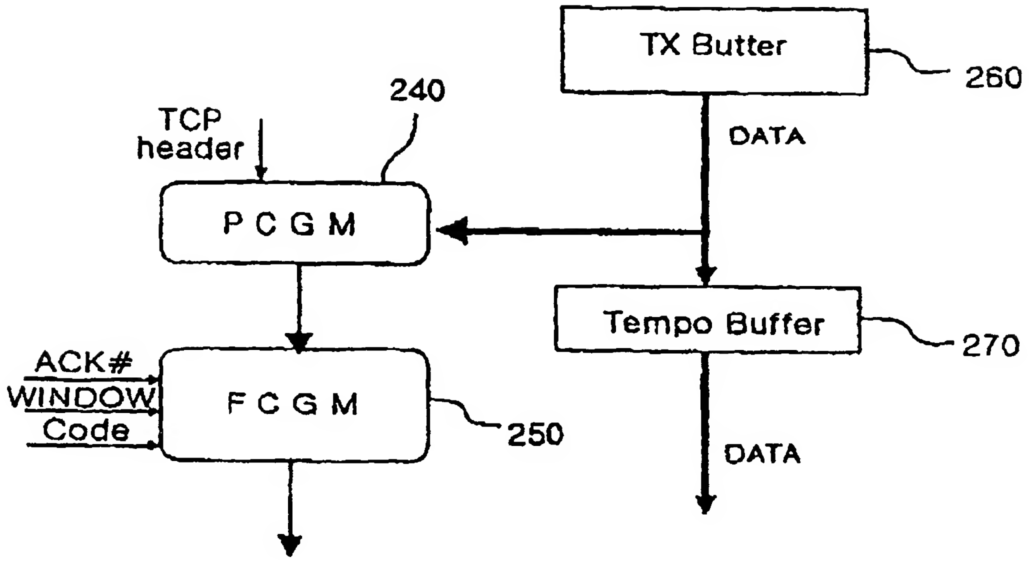

TX Butter
260
DATA
240
TCP header
P C G M
Tempo Buffer
270
ACK#
WINDOW
Code
F C G M
250
DATA

FIG.8

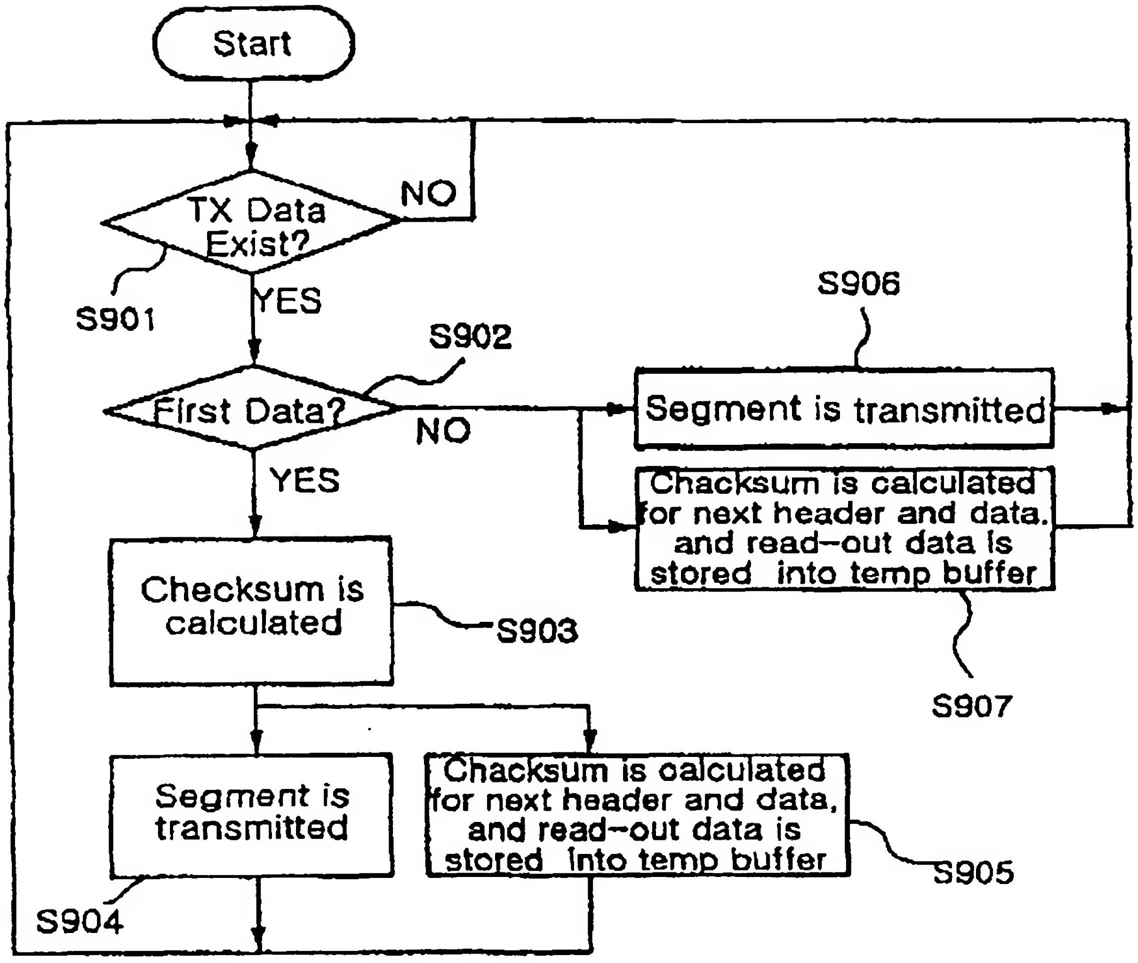

Start
TX Data Exist?
NO
YES
S901
First Data?
S902
NO
YES
Checksum is calculated
S903
Segment is transmitted
S904
Chacksum is calculated for next header and data, and read-out data is stored into temp buffer
S905
S906
Segment is transmitted
Chacksum is calculated for next header and data. and read-out data is stored into temp buffer
S907

FIG.9

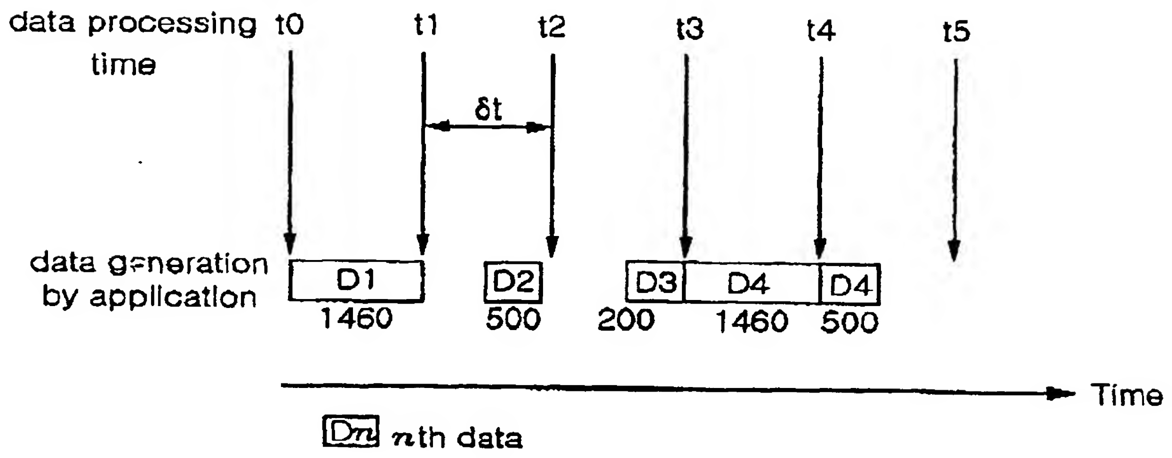

data processing time
t0
t1
t2
t3
t4
t5
δt
data generation by application
D1
1460
D2
500
D3
200
D4
1460
D4
500
Time
Dn nth data

FIG.10

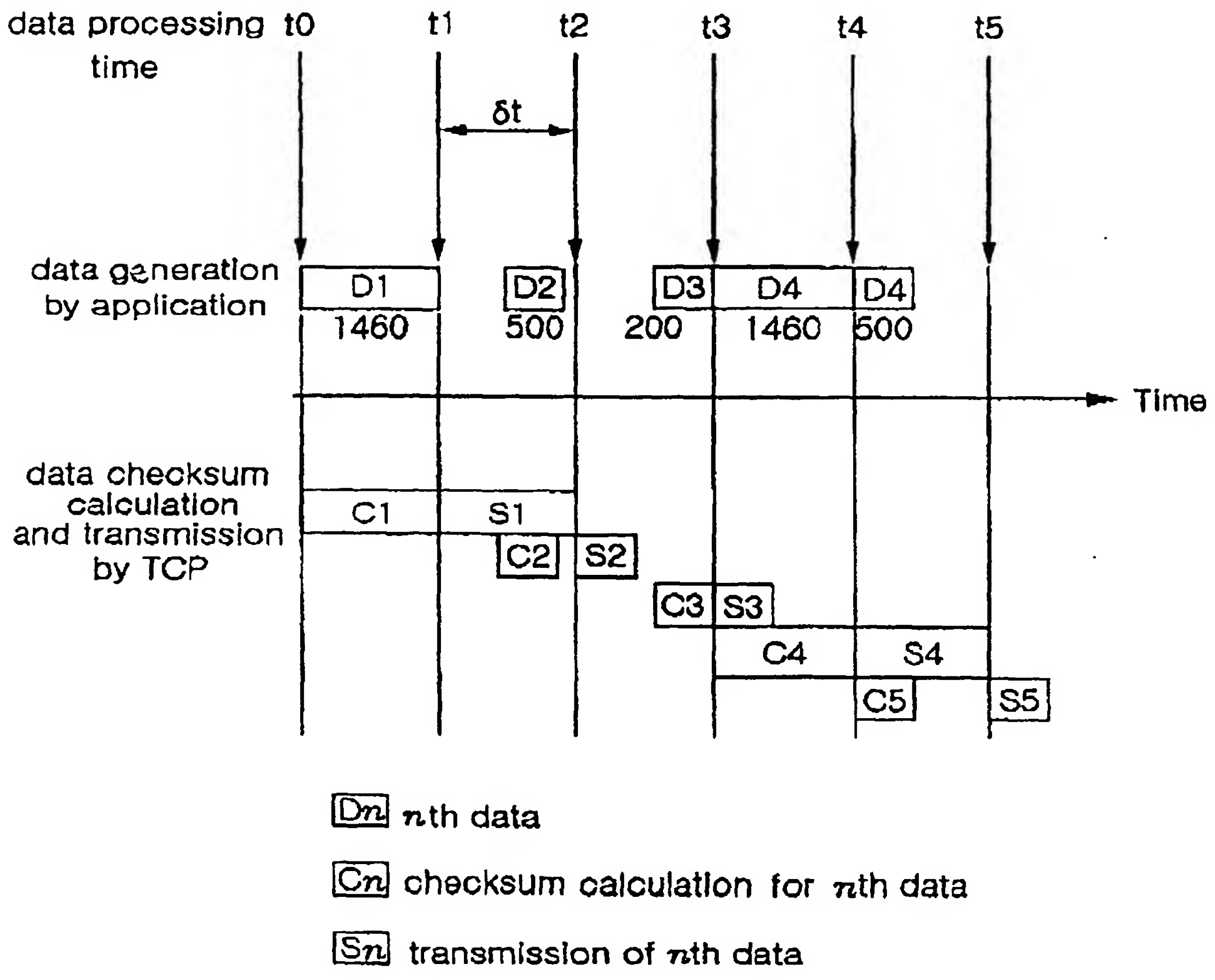

data processing time
t0
t1
t2
t3
t4
t5
δt
data generation by application
D1
1460
D2
500
D3
200
D4
1460
D4
500
Time
data checksum calculation and transmission by TCP
C1
S1
C2
S2
C3
S3
C4
S4
C5
S5
Dn nth data
Cn checksum calculation for nth data
Sn transmission of nth data

FIG.11

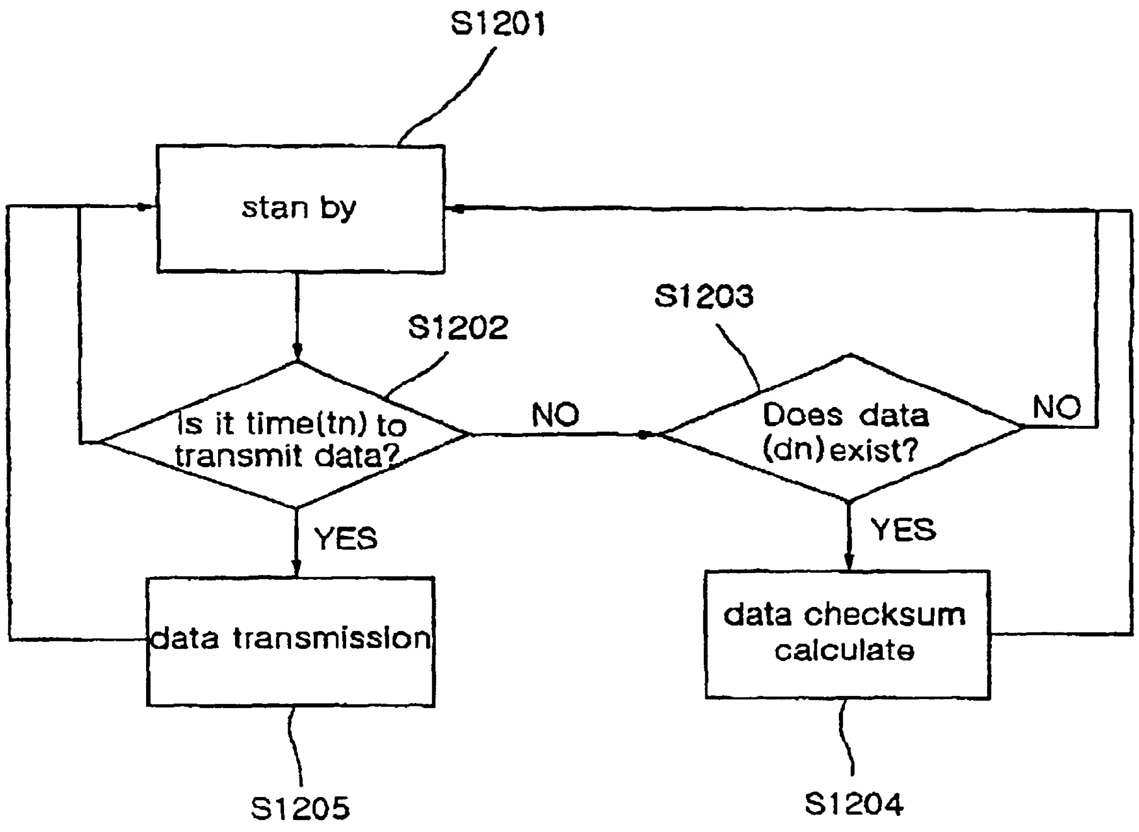

S1201
stan by
S1202
Is it time(tn) to transmit data?
NO
S1203
Does data (dn)exist?
NO
YES
YES
data transmission
data checksum calculate
S1205
S1204

FIG.12

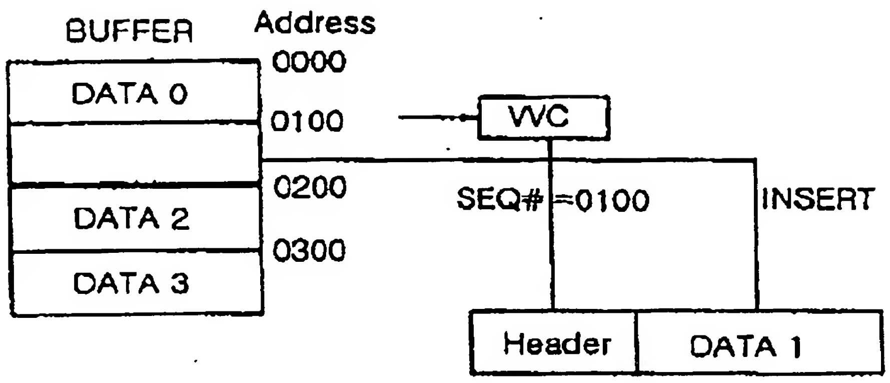

BUFFER
Address
0000
DATA 0
0100
0200
DATA 2
0300
DATA 3
WC
SEQ# =0100
INSERT
Header
DATA 1

FIG.13

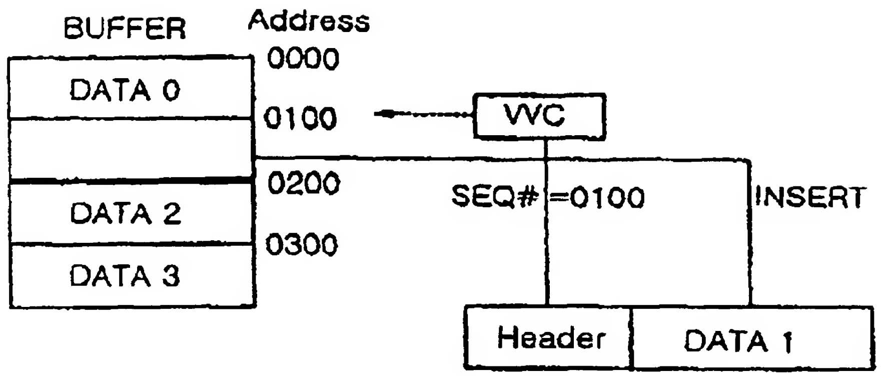

BUFFER
Address
0000
DATA 0
0100
VVC
0200
DATA 2
0300
DATA 3
SEQ#
=0100
INSERT
Header
DATA 1

FIG.14

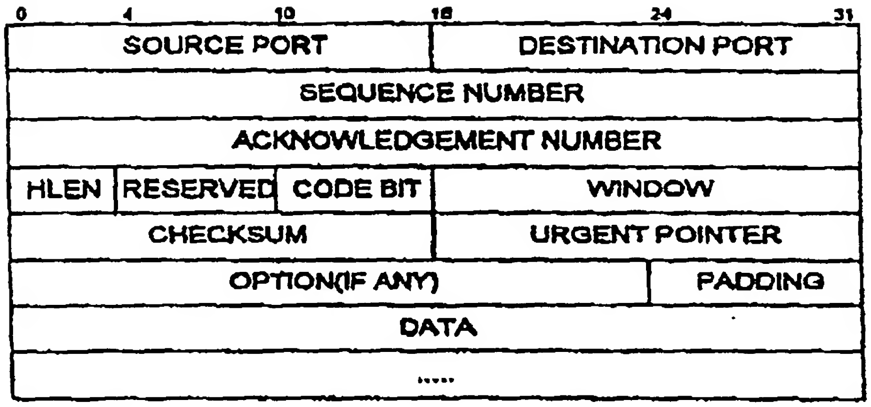

0
4
10
16
24
31
SOURCE PORT
DESTINATION PORT
SEQUENCE NUMBER
ACKNOWLEDGEMENT NUMBER
HLEN
RESERVED
CODE BIT
WINDOW
CHECKSUM
URGENT POINTER
OPTION(IF ANY)
PADDING
DATA
......

FIG.15

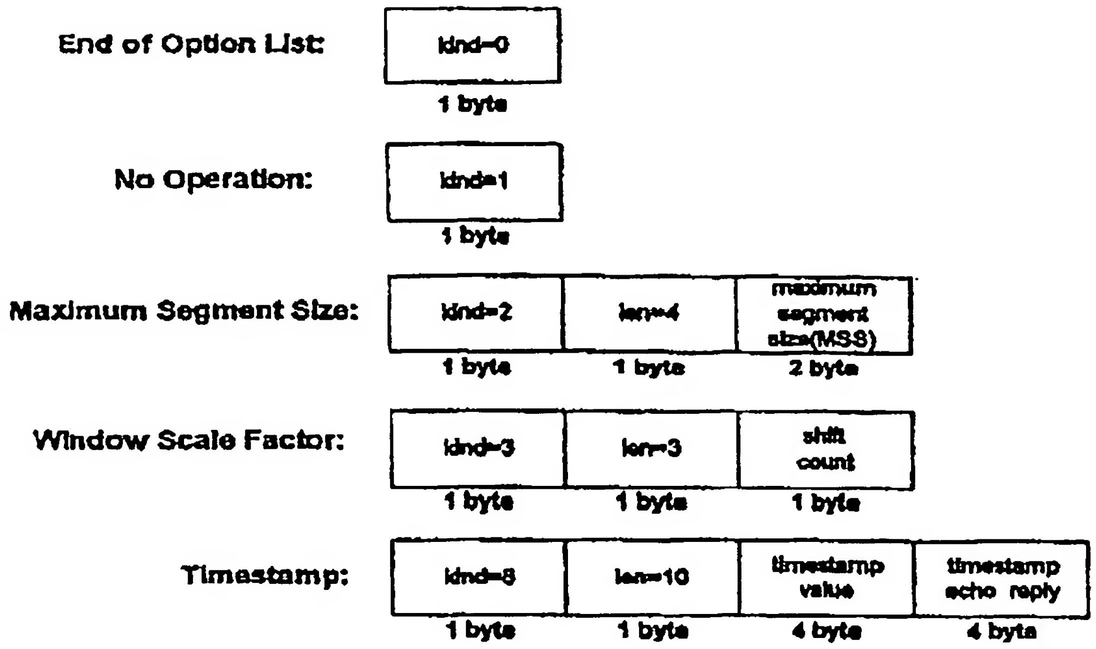

End of Option List:
kind=0
1 byte
No Operation:
kind=1
1 byte
Maximum Segment Size:
kind=2
len=4
maximum segment size(MSS)
1 byte
1 byte
2 byte
Window Scale Factor:
kind=3
len=3
shift count
1 byte
1 byte
1 byte
Timestamp:
kind=8
len=10
timestamp value
timestamp echo reply
1 byte
1 byte
4 byte
4 byte

FIG.16

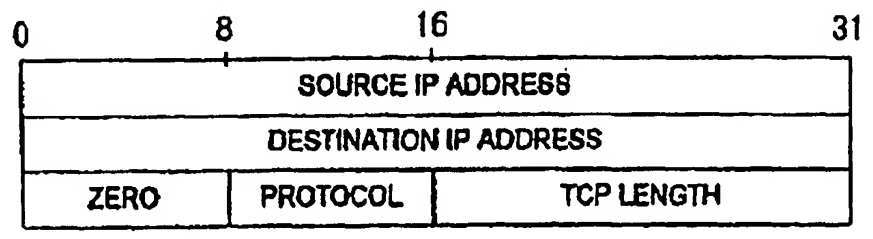
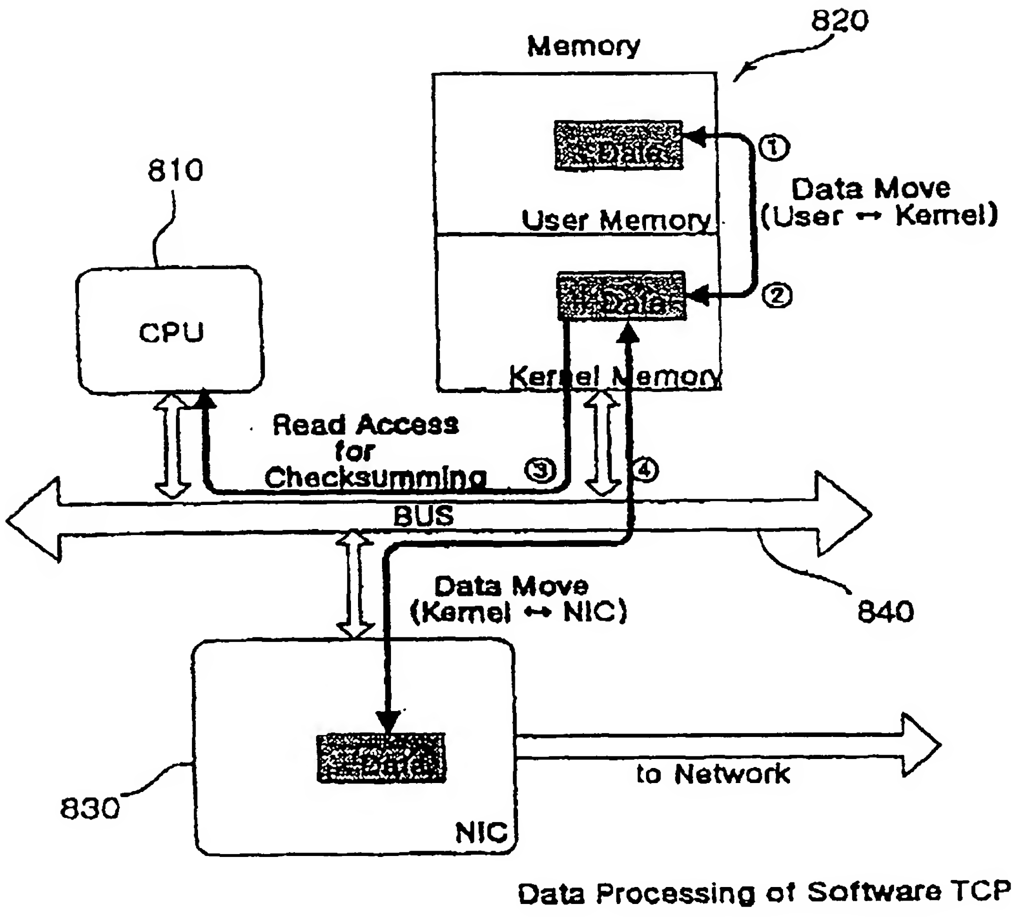

0
8
16
31
SOURCE IP ADDRESS
DESTINATION IP ADDRESS
ZERO
PROTOCOL
TCP LENGTH

FIG.17



820
Memory
Data
①
Data Move
(User ↔ Kernel)
User Memory
810
CPU
Data
②
Kernel Memory
Read Access
for
Checksumming
③
④
BUS
Data Move
(Kernel ↔ NIC)
840
Data
to Network
830
NIC
Data Processing of Software TCP

FIG.18

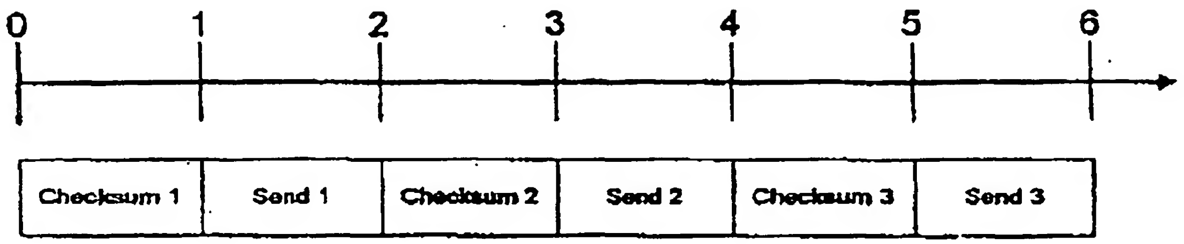


FIG. 19a

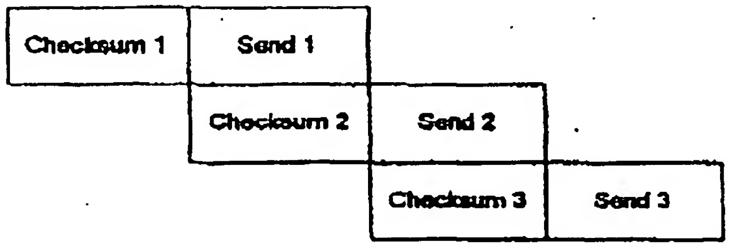


FIG. 19b

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 124 362 A3

(12) EUROPEAN PATENT APPLICATION

(88) Date of publication A3:
07.01.2004 Bulletin 2004/02

(51) Int Cl.7: **H04L 29/06**

(43) Date of publication A2:
16.08.2001 Bulletin 2001/33

(21) Application number: **01300501.2**

(22) Date of filing: **19.01.2001**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **19.01.2000 US 176971 P**
27.07.2000 US 626687

(71) Applicants:
- **Wiznot Corp.**
Kangnam-ku, Seoul (KR)
- **Lee, Jung Tae**
Haewundae-ku, Busan-si (KR)

(72) Inventors:
- **Lee, Jung Tae**
Haewundae-ku Busan-si (KR)
- **Kim, Ku Hwan**
Donglae-ku, Busan-si (KR)

(74) Representative: **Powell, Timothy John**
Eric Potter Clarkson,
Park View House,
58 The Ropewalk
Nottingham NG1 5DD (GB)

(54) Apparatus for processing TCP/IP by hardware, and operating method therefor

(57) An apparatus (100) for processing the TCP/IP (Transmission Control Protocol / Internet Protocol) by hardware is disclosed. In this apparatus, an information plane for transmitting, receiving and processing the user data is separated from a control plane for controlling the transmission and reception of the data by using a hardware. Thus the data transmission is carried out regardless of the processing of the control information. Further, a buffer is provided to store the data to be transmitted and the data received, thereby reducing the number of the memory access times. Further, during the transmission of a segment, the calculation of the checksum for the next predictable header and data is carried out simultaneously, thereby reducing the overhead for the checksum calculation. Further, the addresses of the data to be transmitted are inserted into the sequence number field when transmitting the data, while during the reassembling, the values of the sequence number fields are used as the pointers for storing the data into the memory, thereby improving the overhead related to the memory addressing.

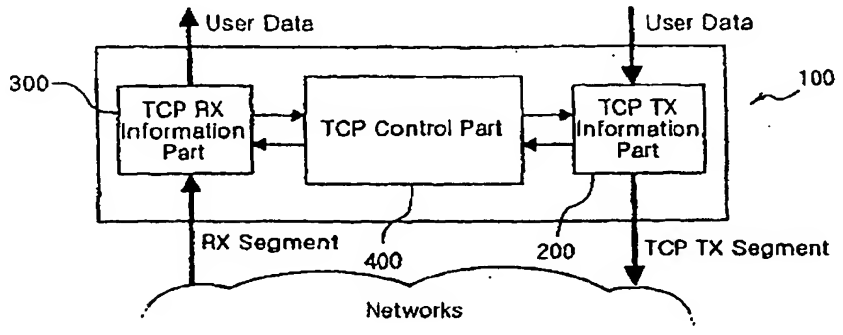


FIG.1

EP 1 124 362 A3

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 01 30 0501

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
X	WO 99 04343 A (INTERPROPHET CORP ;LAWSON MATTHEW TODD (US); JOLITZ LYNNE GREER (U) 28 January 1999 (1999-01-28)	1-4,9,10	H04L29/06
Y	* page 3, line 29 - page 5, line 14 *	11,12	
A	* page 10, line 26 - page 17, line 12; claims 1-10; figures 3,4 *	5-8	

X	US 5 802 278 A (ARUNKUMAR NAGARAJ ET AL) 1 September 1998 (1998-09-01)	1,3,9	
Y	* column 11, line 7 - column 11, line 42 *	11,12	
A	* column 12, line 57 - column 13, line 25 *	2,4-8,10	
	* column 16, line 61 - column 17, line 18 *		
	* column 26, line 54 - column 27, line 54; claims 1-11,18-32; figures 6,8,16 *		

A	WO 97 27539 A (SUN MICROSYSTEMS INC) 31 July 1997 (1997-07-31) *abstract*	1-12	
	* page 8, line 7 - page 13, line 12 *		TECHNICAL FIELDS SEARCHED (Int.Cl.7)
	* page 17, line 6 - page 18, line 3 *		H04L
	* page 43, line 29 - page 44, line 31; claims 31-88; figures 1A,B,2,4A-D,6 *		G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	27 October 2003	Schwibinger, H-P

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons
& : member of the same patent family, corresponding document

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 01 30 0501

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report.
The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

27-10-2003

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9904343 A	28-01-1999	AU 8490898 A	10-02-1999
		WO 9904343 A1	28-01-1999
		US 6173333 B1	09-01-2001
US 5802278 A	01-09-1998	US 5592622 A	07-01-1997
		AU 5736696 A	29-11-1996
		GB 2316843 A ,B	04-03-1998
		GB 2344498 A ,B	07-06-2000
		GB 2344499 A ,B	07-06-2000
		WO 9635988 A1	14-11-1996
		US 5828835 A	27-10-1998
WO 9727539 A	31-07-1997	DE 69713400 D1	18-07-2002
		DE 69713400 T2	31-10-2002
		EP 0976034 A1	02-02-2000
		EP 0976050 A1	02-02-2000
		EP 0976029 A2	02-02-2000
		EP 0976030 A1	02-02-2000
		JP 2000515269 T	14-11-2000
		JP 2000513464 T	10-10-2000
		JP 2000501217 T	02-02-2000
		JP 2000515270 T	14-11-2000
		WO 9727536 A1	31-07-1997
		WO 9727539 A1	31-07-1997
		WO 9727544 A1	31-07-1997
		WO 9727537 A2	31-07-1997
		US 2003115238 A1	19-06-2003
		US 6026485 A	15-02-2000
		US 6014723 A	11-01-2000
		US 6125439 A	26-09-2000
		US 6532531 B1	11-03-2003
		US 5925123 A	20-07-1999
		US 6038643 A	14-03-2000
		US 5970242 A	19-10-1999
		US 6065108 A	16-05-2000
		US 6021469 A	01-02-2000
		US 6076141 A	13-06-2000

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82